DScript Documentation

Installation

##NewDark+Squirrel Installation##
1. Be sure to have at least NewDark 1.25
2. Beside new_dark.zip and editor.zip inside the NewDark1.25.zip open contrib.zip and export squirrel.osm into your main folder.
3. If not present create a folder named *sq_scripts* in your main folder.

##Script Installation##
1. Download DScript.nut (Forum post at ttlg.com)
2. Extract the contents of the .zip directly into your main folder or the DScript.nut into your sq_scripts folder
3. In the editor script_load squirrel

Conventions used:

- Parameters you can define for a script are marked blue.
- Most parameters have a <u>Default Value</u>, which is used if no other value is specified, these are underlined.
  As an example most scripts react to the normal <u>TurnOn</u> message but you can change these to your liking.

  <u>Default Values</u> that are underlined with dashes can not be changed and have to be used/accepted as they are. For example the <u>ResetCount</u> message for counters.
- Bold text like **Editor->Design Note** refer to a property set on an object with the arrow(s) before it guiding you to the location where to find it inside the Add property menu.
- [ScriptName] is the name of the script you are currently reading which should be the same exactly as you entered it into the **S->Scripts** property. A [On/Off] behind the [ScriptName][On/Off] means an optional additional parameter which let's you differentiate further between On or Off messages.

- Orange Squirrel Notes are information only useful if you want to write your own squirrel scripts and use my DScript.nut file as a base. If you are using the scripts only for the editor/in game you can ignore these passages.
- Usable as Root->Function behind a script name means this script can be used as a root script via extends [ScriptName] making the named Functions available.

This is the first draft of the documentation and so not finally revised I recommend to use it together with the documentation inside the the DScript.nut file.

DBaseTrap (or how to use DScripts)

Every script here – if not especially said to be not a DBaseScript - uses the syntax and features of this ‘Trap’ as a foundation. It allows you to set parameters for various functionality.
Every parameter explained here has to be put into the **Editor->Design Note** separated from the next parameter via a semicolon **;**.

New Dark feature:
Clicking with the middle mouse button on the edit control in the "Design Note" / "Objlist Arg" property dialogs will open a special param edit dialog. (Click here for the full explanation)

[ScriptName]On and **[ScriptName]Off**
Normally scripts react to TurnOn and TurnOff messages to perform their On respectively their Off action.
With *[ScriptName]On/Off="YourMessage"* you can define other messages this script will react to.

DScripts can react to multiple TurnOn/Off messages.
In such a case each message needs a **+** before them like: [ScriptName]On="+TurnOn+FrobWorldEnd"
Here the script will react to a TurnOn message as well as a FrobWorldEnd (release of the right mouse button)

Squirrel Note: It is possible for a script to have another On/Off default message. You can specify this with *DefOn="CustomTurnOn"* or *DefOff="YourOff"* anywhere in your script class but outside of functions. Messages specified in the Design Note have priority. DWatchMe is currently the only exception which uses *BeginScript* instead of TurnOn.

As a matter of fact this lets you simulate the three **Trap Control Flags** *NoOn, NoOff, Invert* – NOTE: The standard Trap Control Flags set with the object property have no effect on DScripts!

You can use *[ScriptName]On="TurnOff";[ScriptName]Off="TurnOn"* to simulate *Invert*; *[ScriptName]On="Null"* can be used to simulate *NoOn* and *[ScriptName]Off="Null"* can be used to simulate *NoOff*.
Remember that if you specify *[ScriptName]On="TurnOff"* without specifying *[ScriptName]Off*, then TurnOff will perform the On as well as the Off action.

[ScriptName]Count
Use this parameter to specify the maximum number of times that the script will work. You can use [ScriptName]Count=1 to emulate the Once Trap Control Flag. The default is 0 (infinite).
Both TurnOn and TurnOff are counted by default. You can use [ScriptName]CountOnly=1 to count only TurnOn, and [ScriptName]CountOnly=2 to count only TurnOff - the respectively other message will still have their full effect even if the Count is full. Sending the trap a ResetCount message will reset the counter.

[ScriptName][On/Off]Capacitor
Use this parameter to specify the number of times that a trap will need to receive its triggering message *before* it activates. For

example, a DRelayTrap with DRelayTrapCapacitor=3 will only relay every third message, while one with NVRelayTrapOnCapacitor=4; DRelayTrapOffCapacitor=2 will relay every second TurnOff message but only every fourth TurnOn message. The default is 1.
[ScriptName][On/Off]CapacitorFalloff can be used to specify the time, in [seconds], that it takes for the trap to "lose charge", and the activation count to go back down by one activation.

NOTE 1: While testing!:Count and Capacitor data are refreshed on script_reload and *exiting* game_mode. Therefore I suggest that you insert the script_reload command into your GameMode.cmd to run it and update the data before entering game mode.
NOTE 2: All Capacitor parameters don't interfere with each other. For example setting Capacitor=3 and OnCapacitor=2 and then receiving TurnOff,TurnOff,TurnOn,TurnOn will relay both TurnOn messages as the first TurnOn is the 3rd message and the 2nd is the 2nd TurnOn.
Same goes for CapacitorFalloff it won't decrease an OnCapacitor only OnCapacitorFalloff will do.

[ScriptName]Delay
Setting this to a value greater 0 will postpone the action performed by this script by the specified number in *seconds*.
[ScriptName]Delay=0.5 for example would delay the action by 500ms.
By setting [ScriptName]ExlusiveDelay=1 the script will cancel the other activations which still are on delay and will start the delay again at 0. This is for example useful if you want a script to react to Water Arrows or entering a Room/OBB where due to the engine multiple On messages are sent in a very short time.

[ScriptName]Repeat
If you want a script to repeat its actions multiple times you can set this parameter. By default it is 0 for none. The time between repeats is specified with [ScriptName]Delay.
Infinite repeats are declared via [ScriptName]Repeat=-1 and will be stopped by the corresponding other message. A infinite repeating TurnOn will be stopped by a TurnOff. This single TurnOff will only stop the repeat and will *not* perform the DoOff action of the script. Only a second TurnOff will trigger it.

[ScriptName]FailChance
A chance (from 1 to 100) that the specific script will NOT activate.
If you define a negative value (-1 to -100) Capacitors and Counts will still be increased first and then it will tested if the script will fail or not. A set Count and a negative fail chance could lead to a situation where the script never activates.

[ScriptName]Target and **[ScriptName]TDest**
If a script is looking for an object parameter you can use the following expressions.
Further $QVar can be used for all Parameters as it represents a number for example DrelayTrapRepeat="$MyQVar"
[ScriptName]TDest is only available for message sending scripts which extend from the DrelayTrap script.

ObjectName	The specific name of an object. Hint: The Player object is named Player.
ObjID	The specific (object ID)
[me]	The object itself where the script is placed.
[source]	The object that sent the triggering message.
&LinkType	All linked objects with the specific link type.
*Archetype or Meta-Property	All concrete objects which are based from exact this Archetype or having exactly this Meta-Property
@Archetype or Meta-Property	Same as above but including descending archetypes and Meta-Properties as well.
^Archetype or MetaProp or Name	Finds the closest object with this name.
$QVar	The Object with the ID stored in the Quest Variable QVar
+Anything	Combine different sets of objects in combination with the above characters.
+-Anything	Removes a subset from previously added objects

The + Operator

IMPORTANT: When using the + operator you need a # before object/archetype/MetaProp ID numbers.
Like: "+#42+#-17"

Example:

- [ScriptName]Target="+@guard+-@M-FrontGateGuard+-@M-TopFloorGuard"
  This perform the action on all guards, excluding all which have the M-FrontGateGuard or M-TopFloorGuard meta property.

- [ScriptName]Target="+[me]+&~ControlDevice+coolguy+#42"
  This will give back the object itself, all ~ControlDevice'd linked objects, the object named coolguy and the object with the number 42.

Note: Even if only one object is needed like for DCopyPropertySource="&LinkFlavour" can still be used. It will only return one object. It will be the one with the highest LinkID. (Needs Confirmation)

Tip:
From my testings Scripts from different .nut files are shared, so you could also use extend DBaseTrap in your files without having to creating a new class yourself. Only restriction is that your file needs to be alphabetically behind the DScript.nut

---

DRelayTrap **Can be used as RootScript -> DSendMessage; DRelayMessages**

A RelayTrap with all the DBaseTrap features. On TurnOn it will relay the messages specified with DRelayTrapTOn. Respectively on TurnOff the messages specified with DRelayTrapTOff are. By default TurnOn and TurnOff are being sent.
With the + operator you can define multiple On, TOn, Off and TOff messages.
With DRelayTrap[On/Off]Target you can specify where to sent the message(s) to. Default are &ControlDevice linked objects. If DRelayTrap**On**Target is specified then it will take priority over DRelayTrapTarget. You can target different objects for the TurnOn/Off messages.
DRelayTrap[On/Off]TDest is also a valid alternative option to be more similar to NVScript.

A Stimulus can also be sent to do this, first enter the intensity surrounded by square brackets, followed by the stim name. For example: [ScriptName]TOn="[5.00]WaterStim".

Design Note example:
NVRelayTrapOn="+TurnOn+BashStimStimulus";NVRelayTrapTOn="+TurnOn+[5]FireStim";NVRelayTrapOnTarget="+player+^ZombieTypes"

What will happen:
On TurnOn or when bashed it will send a TurnOn and a FireStim with intensity 5 to the player and the closest Zombie.(relative to the object with the script)
As nothing else is specified on TurnOff will send a TurnOff to all ControlDevice linked objects.

DHub **(not a DBaseTrap script)**

Very Similar to the DRelayTrap but it can perform different actions on different inputs.

Will by default send a TurnOn at ControlDeviced linked objects when it receives a valid message.
Valid messages are declared in the DesignNote like this: DHub[Message]="Parameter1=Value1;Parameter2=Value2". Even if you want to use the default options a minimal DHub[Message]="==" is necessary! The + operator does not work for valid massages.

Multiple actions can be declared for the same Message via DHub[Message]2="Parameter1=Value1;.... The used numbers must be consecutive! If there is a gap all above numbers will be skipped. There is no maximum to different actions a valid message can have.

Parameters are basically identically to the ones by **DBaseTrap** or **DRelayTrap** beside [T]On/Off.
NOTE:!Parameters require no prefix. String values may not have " " around them!!! And I don't recommend blank spaces anywhere inside the "Parameter=Value;P2=V2" declaration, only if you want to address an object like Target=@false label

List of valid parameters – all operators like +,@,$... will work here
Code:

Relay	What message(s) should be sent. Like for DRelayTrap there is stim support like [3.7]FireStim.
Target	To target(s)
Delay	Will delay the message by the specified amount in seconds and also defines the time between repeats.
ExclusiveDelay	When set to =1 will abort messages from this action which are still waiting to be sent.
Repeat	How often shall the action be repeated. Repeat=-1 for infinitely. *All* Infinite repeats can be stop via a StopRepeat Message. A single repeat (which can also be a specific number) via [Message+number]StopRepeat.
Count	The maximum number of times an action will be executed. Default =0 infinitely. *All* counts can be reset via a ResetCount message. Single action counts like above with via [Message+number]ResetCount.

Capacitor	The number of times this action will need to receive its valid massage before executing.
CapacitorFalloff	The time in seconds after the Capacitor threshold will go down by 1.
FailChance	A chance (from 1 to 100) that this action is not executed. If the number is negative Capacitor and Counts will still be increased first and then tested if the script will fail or not.

Default Parameter:
By setting DHub[Parameter]="Value". This Parameter will be default for all actions if not specified otherwise.
As an example and as script default DHubRelay="TurnOn";DHubTarget="&ControlDevice".

Design Note example
https://www.dropbox.com/s/nzwt7tzhxg...mple.jpg?raw=1

DCopyPropertyTrap

Target default: &ScriptParams
Source default: [me]

Similar to S&R->SetProperty but can set multiple properties on multiple objects at the same time.

Upon receiving TurnOn copies the properties specified by DCopyPropertyTrapProperty form the object.
DCopyPropertyTrapSource (default is [me]) to the objects specified through DCopyPropertyTrapTarget if not set ScriptParams linked objects will receive the propertie(s) by default.
Multiple properties can be copied with the + operator.

You can use the script object as a sender or receiver.

Design Note example:
DCopyPropertyTrapProperty="+PhysControl+RenderAlpha";DCopyPropertyTrapSource="&Owns";DCopyPropertyTrapTarget="[me]"
This will copy the Physics->Controls and Renderer->Transparency(Alpha) property from the object linked with an Owns linked to the object itself.

DWatchMe

By default when this object is created or at game start (BeginScript) creates AIWatchObj Links from all Human(-14) to this object. Alternatively an alternate DWatchMeOn message can be used, as well as DWatchMeTarget to specify other objects.

•Further (if set) **copies!** the **AI->Utility->Watch links default** property of the archetype (or the closest ancestors with this property) and sets the **Step 1 - Argument 1** to the Object ID of this object.
•Alternatively if no ancestor has this property the property of the script object will be used and NO arguments will be changed. (So it will behave like the normal T1/PublicScripts WatchMe or NVWatchMeTrap scripts).

On TurnOff will remove any(!) AIWatchObj links to this object. You maybe want to set DwatchMeOff="Null".

Usefullness:
If you have multiple objects in your map and want that AIs perform simple actions with each of them under certain conditions.

For example:
(Check the Demo:)
You can use it to let guards relight every extinguished torches in their patrol path.

Tip:
If you use a custom On command, multiple links could be created so you may want to set the Watch->"Kill like links" flag in that situation.

Design Note Examples:
DWatchMeOn="WaterStimStimulus";DWatchMeTarget="@guard";DWatchMeFailChance=60
DWatchMeOn="TurnOn";DWatchMeTarget=42
DWatchMeTarget=ValidObjectName

DStdButton **(DRelayTrap)**

DStdButton has all the StdButton features- even TrapControlFlags work. Once will lock the Object - as well as the DRelayTrap features, so basically this can save some script markers which only wait for a Button TurnOn.

Additional:

If the button is locked the joint will not activate and the Schema specified by DStdButtonLockSound will be played, by default "noluck" the wrong lockpick sound.

DAddScript Can be used as Root -> D[Add/Remove]ScriptFunc

Adds the Script specified by DAddScriptScript to the objects specified by DAddScriptTarget, by default &ControlDevice. Additionally it sets the DesignNote via DAddScriptDN. If Script parameter is not set only the DesignNote is added/changed.

On TurnOff will clear the **S->Script: Script 4** slot.

NOTE:
• It will try to add the Script in Slot 4. It will check if its empty or in case if it's not it got copied from the Archetype, else you will get an error and should use a Metaproperty.
• It is possible to only change the DesignNote with this script and so change the behavior of other scripts BUT this only works for non-squirrel scripts!
• Using Capacitor or Count for will not work for newly added DScripts. As these are created and kept clean in the Editor.

DArmAttachment

Attaches another object to the players arm.

When using an empty hand model you can create your own custom weapons. It can be given to any weapon-ready object together with **Inventory-> Limb Object: emptyhan or BJACHAND** (Both Models included, made by Jason Otto and Soul Tear). OR you can also add it to normal weapons to attach a Shield or Light or whatever creative stuff you come up with ;)

Parameters:

DArmAttachmentUseObject:
= 0 (Default): Will create a dummy object and give it the **Shape->Model**: DArmAttachmentModel
= 1 Will create a copy of a real object/archetype specified in DArmAttachmentModel. If you want to attach special effects for example.
= 2 (experimental and not really working): Same as 1 but the object will be really physical -> Real collision sounds based on the object.
= 3 (experimental little working): Same as 2 but works even better. Disadvantage: Errors in DromED, Model will pass through walls/objects.

DArmAttachmentModel: has to be a model file name(0) or an object Archtype (1) depending on (DarmAttachmentUseObject)

DArmAttachmentRot and DArmAttachmentPos:
The model will most likely need some adjustments in position and rotation. Both parameters take 3 arguments (a vector) separated by commas, like this: "90,0,0" or "0.1,-0.6,-0.43". They stand for HPB rotation and XYZ translation. But most like don't behave like you expect it.

TIP: Best method how to figure out the numbers is to use it in combination with set game_mode_backup 0 (**NEVER SAVE AFTER!**) and modify the DetailAttachment Link. It's trial and error.

NOTE: You will also need to do some Hierarchy changes to adjust the sound and motions. Also I would say this script is not 100% finished please give feedback.
TIP: Remember you can use multiple different melee weapons by using the cycle command. Or design them so that you can also use them from the normal inventory.

https://www.dropbox.com/s/htdofcgz71...tach.gif?raw=1

DHitScanTrap (DRelayTrap)

(Has the DRelayTrap sending features: [On/Off]Target or [On/Off]TDest)

When activated will scan if there is an object or a wall between two objects. Imagine it as an invisible scanning laser beam. The start and end object are specified by **DHitScanTrapFrom** and **DHitScanTrapTo**, the script object is used as default if none is specified.
If the From object is the player the camera position is used, if the To object is the player as well then the ray will follow the center of the players view – for example to check if hes exactly facing something.

The Object that was hit will receive the message specified by DHitScanTrapHitMsg (default is *DHitScan*). By default when any object is hit a *TurnOn* will be sent to ControlDevice linked objects. Of course these can be changed via DHitScanTrapTOn and DHitScanTrapTDest
Alternatively if just a special set of objects should trigger a TurnOn then these can be specified via DHitScanTrapTriggers. **THIS WAS CHANGED in 0.28a from Target to Trigger!!!**

NOTE: Unlike other scripts DHitScanTrapFrom and To only take single objects at the moment. I can add multiple object support as well but want to do some performance test first.
NOTE2: The scanning is described as a rather expensive progress. So don't use it like 100 times per second over long distances. Best to only activate it when you need it. (That's why this is no real Trigger)

I suggest to use DhitScanTrapRepeat=-1 in combination with a Room script.

DRay

This script creates and scales particle effects between objects specified by DRayFrom and DRayTo with the + or similar operator you can create multiple and different SFX between multiple objects at once.
It works with a template particle effect defined by DRaySFX which can be a group of special effects as well.

Sending further TurnOn messages will not create additional particle objects but update the already present ones.

There are two different types of effects that can be used

DRayScaling: = 0 (Default): For evenly stretched particle effects. This will scale up their spawn box and the particles used. A small effect with look exactly like a big one. = 1: For "shooting" Particle effects which start at the From location. Increases not the number but the lifetime of the particles (Min=Max).

NOTE: For all effects only the X values and Particle Launch Infos are used!

TIPS:
• A good template is the ParticleBeam(-3445)
• You can use this script directly in the editor to automatically set up effects with an appropriate DRayOn message. For example "test" (use script_test objId) or Sim (objects will be created after leaving game mode) and then delete the script object.
• By using **Particles Start Launched** you can create a different and more solid effect but they will be less accurate in hitting the To object.
• You can either use an infinite repeating TurnOn from an outside Trap or DRayRepeat=-1 to constantly update your special effects if your objects are moving.
• If you don't want to turn off the special effect the second method is better else I recommend the first. (Else you would need +TurnOff+TurnOff+TurnOn sequence to deactivate the script completely.

• When using launched very fast particles (like the laser in the video) the effect degenerates after a few seconds, so sending a "+TurnOff+TurnOn" with DRayOffCapacitor=50 will update the effect 50 times and then create a new one. The capacitor doesn't have to be that high but with a low/no capacitor very much performance is used to calculate the launched particles! (additionally it's smoother as well)
• My Demo effects are rather simple, be creative with the particle settings, share your stuff if you for example created a spiral or or or...:D

All paricles are created with DRay and the damage detection with DHitScan<video controls="controls" height="480" width="640"><source src="http://dl.dropboxusercontent.com/s/kr3nxz9mlm965p3/Dark%20Projekt%2008.01.2017%20-%2000.15.19.09.mp4?dl=0" type="video/mp4"></source></video>

DDumpModels

(Included in the DDumpModel.nut read the instructions for more detailed information. Or in this post here)

Ever wondered what sort of models you accumulated in your obj folder?

This script will create all the models from your obj folder and places them in the editor.
The parameters used are exactly! as written here (no DScript Syntax):

First= the model name of the first model you want to use; if it's a number it will start with for example the 100th model in your list.
MaxModels = will stop after for example 2000 have been created.
Screenshots=1 will automatically create screenshots in game mode while creating the models.

DHudCompass

Upon receiving FrobInvEnd will clone the current selected object and attach it slightly below the players camera and keeps its facing always to north – so this script is not limited to the compass.
By default there is no TurnOff as it is TurnedOff via toggling.

Engine Features -> Frob Info->Inv Action: Script must be set.

I suggest scaling down the object to 10-50%.
https://www.dropbox.com/s/dsp0xu1lc3...pass.gif?raw=1

DDrunkPlayerTrap

On TurnOn makes you drunk a TurnOff sober. The effect slowly gets stronger and after the time is over it will perform a fading out as well.
Multiple TurnOns from the same source will reset the timer and will do a FadeIn again. Multiple sources do stack.

Optional parameters:

- DDrunkPlayerTrapStrength=1 regulates the strength basically every number can be used I would suggest something between 0 and 2.
- DDrunkPlayerTrapInterval=0.2 A second regulator how often the effect is applied in seconds. With a lower intervals higher strength becomes more acceptable.
- DDrunkPlayerTrapLength=0 How long the effect will last in seconds. Use 0 for until TurnOff is received.
- DDrunkPlayerTrapFadeIn=0 Fades in the effect over the given time at the end the full strength is used.default[seconds]=0
- DDrunkPlayerTrapFadeOut=0 **Only works if Length is set**! Will gradually make the effect weaker over the last given seconds.
- DDrunkPlayerTrapMode=3 The effect is made up by 1) shaking the camera and 2) Pushing the player.(left/right/forward). Mode=3 uses both effects.

By setting DDrunkPlayerTrapMode to 1 or 2 you can use one effect only. Especially for Mode=2 higher Strength values can be used.

Check out the demo how this can be used as a potion.
I think you can also create some interesting effects with an early FadeOut and DdrunkPlayerTrapRepeat.

DTPBase

Base script for the other 3 scripts. Has by itself no use.
•If a patrolling AI is teleported it will reassume its patrol after teleportation from the now closest Patrol Point.
•Contains the DTeleportation(who,where) function which teleports a given object to absolute world coordinates.

DTeleportPlayerTrap

•Upon receiving TurnOn moves the player by x,y,z values specified in the Design Note via *DTpX=;DTpY=;DTpZ=*. Use for 100% Seamless teleportation.
•If no parameter is set or all are 0 then the player gets teleported to the location of this object instead.

Design Note Example:
DTpX=-3.5;DTpZ=10

DTrapTeleporter

Target default: &Control Device

Upon receiving TurnOn teleports a ControlDevice linked object to this object and keeps the original rotation of the object.
Further by default currently non-moving or non-AI objects will be static at the position of the TrapTeleporter and not affected by gravity until their physics are enabled again - for example by touching them.
By setting *DTeleportStatic=0* in the Editor->Design Note they will be affected by gravity after teleportation. Does nothing if Controls->Location is set.

Design Note Example:
DTeleportStatic=0
DTrapTeleporterTarget=@Zombie types

DPortal

Default TurnOn: PhysEnter
Default Target: Entering Object. (not [source]!)

Teleports **any entering object** (PhysEnter).

Either by x,y,z values specified in the Design Note via *DTpX=;DTpY=;DTpZ=* or to object linked with ScriptParams. If a value in the Design Note is specified the first option takes priority!
Unlike DTeleportPlayerTrap this script takes the little offset between the player and the portal center into account, which enables a 100% seamless transition - necessary if you want to trick the player.

Design Note Example:
DTpX=-3.5;DTpZ=10
DPortalTarget="+player+#88+@M-MySpecialAIs"
Tip
If you use the ScriptParams link and want a seamless transition, place the destination object ~3 units above the ground

---

DCompileTrap

On TurnOn tries to compile the Editor Comments. NOT Design Note.
This script lets you write your own squirrel commands inside the editor.

It uses Editor Comment because in Squirrel the semicolon **;** indicates a new line in the DesignNote it would be interpreted as a new Parameter. So you are able to use newlines much easier.
NOTE: But the newlines(made by enter) will be translated to \n and to not destroy your code you need to comment these \n out with a /* at the end of the line and */ at the start of the next

Editor Comment example:

```
ActReact.Stimulate(Object.Named("player"),"BashStim",20);/* now comes a new line
*/DarkUI.TextMessage("String made by DCompileTrap");/*
*/DarkGame.KillPlayer()
```

DScript 0.28a Squirrel Features

These features/functions specified here are globally available to who has the DScript.nut file and **are not bound to the DBaseTrap.**

DGetParam("FullParameterName",DefaultValue, Table ,adv=0)
Returns the parameter in the DesignNote if none is present the DefaultValue instead. The 3rd parameter is where you actually look most of the time you will pass the Design Note userparams() on here but also check in every other table-like structure like classes.
if a value is found or else the default it will be analyzed by:

DCheckString(r,adv)
This function analyzed the given string r and performs the operator checks as listed in DBaseScript. @,&,[me],+ ...
Each found object will be stored in an array. If adv = true the whole array will be returned and can be used for foreach (t in array) { ... }. Else if adv=0/false only a single value will be returned.

DGetStringParam("FullParameterName",DefaultValue, String ,adv=0)
Identically to DGetParam but that this function takes strings which are formatted like this: "ParameterName=Value;Key=Value2". DCheckString is then called as well.

DGetAllDescendants(Archetype,array)
Function behind the @operator.
Returns all concrete descendants of an archetype and all other inherited archtypes in an array. The function must be given an array to work with all found objects will be added to that array.

DArrayToString(Array , Seperator ="+") converts an array to a string with the character defined by Seperator between them, default +. The reverse DStringToArray to string split.

DSendMessage**(Target, Message)** (need extends DRelayTrap)
Will send a single *Message* to a single *Target* and check if the *Message* should be a Stim. If so it will stimulate the Target.

DRelayMessages**("On" or "Off", DesignNote)** (need extends DRelayTrap)
Will relay the Message**s** in [ScriptName]T[On/Off] to the objects [ScriptName][On/Off]TDest via DSendMessage.

DCountCapaCheck(Script Name, Design Note, true/false for On/Off) and DCapacitorCheck
These handle partly the Count and Capacitor parameters for the DBaseTrap and then call the DoOn/Off functions. As they are

global functions I noted them here but they will not work as standalone.

DSetTimerData(name, delay,...[data you want to attach]) returns timer_handle
DSetTimerDataTo(To,name,delay,...) same as above but sent to To instead of self.
DgetTimerData(msg.data,KeyValue=false) returns an Array with your data.

The functions start a timer with multiple data attachments which can be returned later in an array out of the message().data.
This Data is save game compatible!

But now let me explain why these functions are important/usefull:
If there time gap between 2 actions, there is always the problem how to safely carry over data. The (human) player could save and reload the game in the meantime and all non persistent data will be whipped as new script instances are recreated
The relevant persistent data are script data set with SetData(k,v), QVars and timers.

I don't know it but I think NV came up with something similar so I take an NVRelayTrap as an example: which inverts the message and sends it twice back to the sender (with a delay).
So now we got two objects sending a TurnOn and TurnOff to the Trap and here's the problem a) after the delay how to know which object sent the message? b) What was the message again? and c) how to know if it's first message that needs to be send again?
We would need to save three values. But SetOneShotTimer only has one data slot. The second message would override the first SetData or QVar. Well you could store multiple ScriptData with a prefix and save that prefix in the timer data slot, but I think memory wise that's not the best solution, also it would increase very fast if you need to transfer even more data.
Btw: Squirrel somehow lacks the SetTimedMessage(2) function why?.

Data can be stored and retrieved in a fixed order (KeyValue=false) or in a not sorted Key=Value manner, for the later see below.
As an example for the sorted manner if you know the first data is always the source and the second the are repeats left. Returned Array will be[source Obj ID, Repeats left]
And you can use array[0],array[1] to grab your data.

If there are other data you want to store but for example some are optional or whatever you can store them via key=value,key2=value2,...
now if you use KeyValue=true the array will look like this [key,value,key2,value2] and you can search and find your values like this:
Code:

```
local array=DGetTimerData(message().data,true)
local key = array.find(param);
loval value =0

if (key!=null)
   {value =array[key+1]}
```

Take a look at the DDrunkPlayerTrap how I made use of that.

Appendix

Clicking with the middle mouse button on the edit control in the "Design Note" / "Objlist Arg" property dialogs will open a special param edit dialog. This dialog is designed for text that is formatted as typical script parameters, in "<param name>=<value>;" format, and displays the params in list/table form. Property text that doesn't comply to this param-value format will prevent the dialog from opening. To edit param names or values, double click on the name or value in the list. It's also possible to press TAB or RETURN to start editing the (first) selected item. You can tab through all elements in the list. To commit an edited name/value you have press RETURN or TAB (or navigate away with cursor UP/DOWN). If there's some formatting error in the value it will fail to commit, and the edit control will remain active for further editing. Pressing ESC or clicking on something else will revert to the original text. To add a new param, start editing the "<<<add new>>>" item that's last in the list (or press INS). Params can be deleted by selecting them and pressing DEL. Copy-paste is supported.
